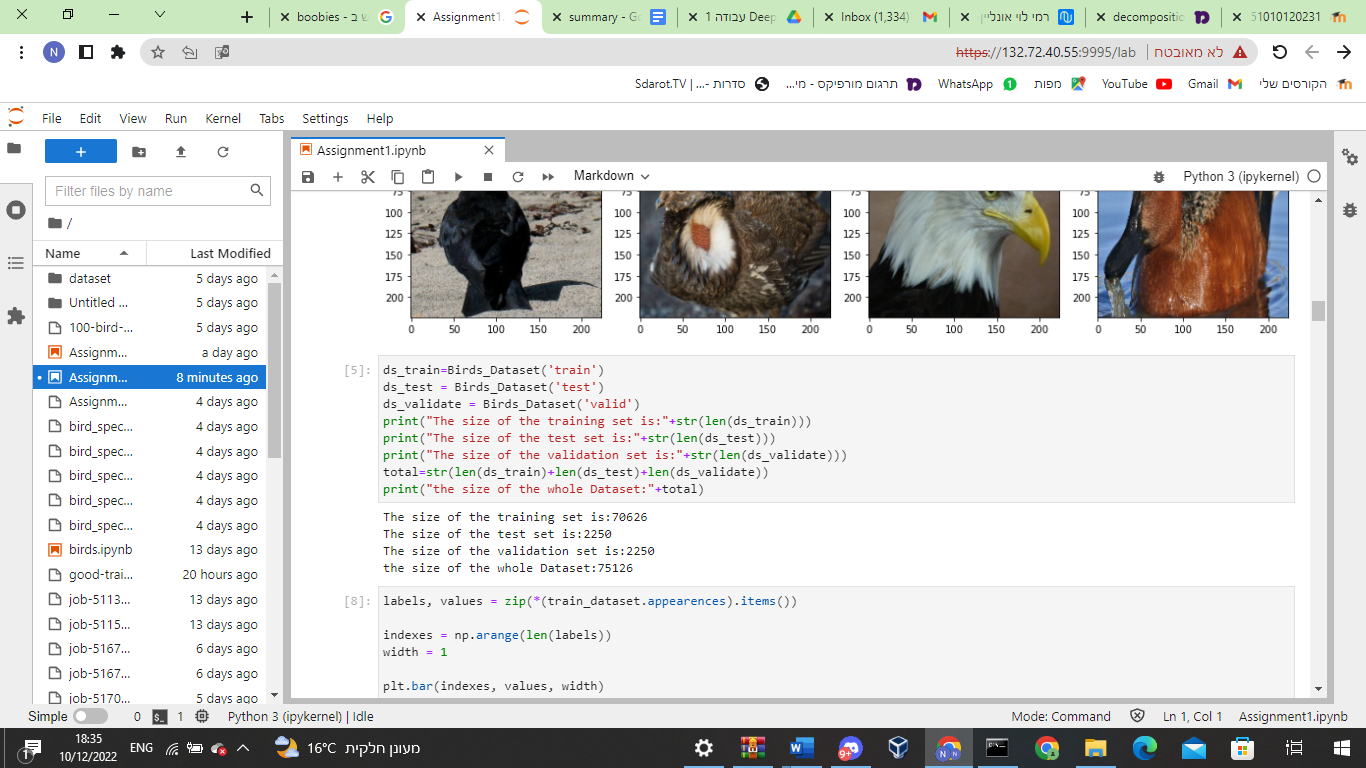
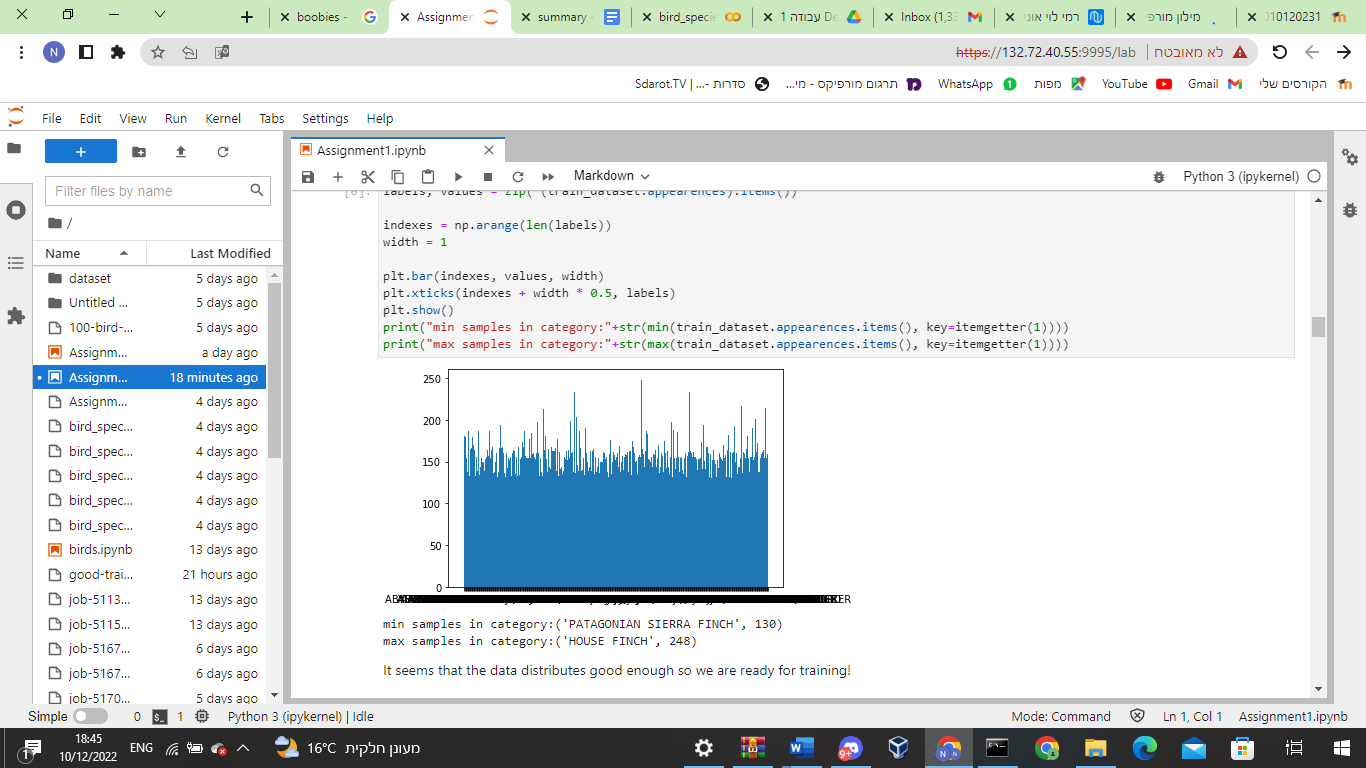
Part 1:

In this research project we would like to create a DNN that classifies bird species.

We used the dataset from [kaggle.](https://www.kaggle.com/datasets/gpiosenka/100-bird-species/code) and in our research we took inspiration from the code examples there.

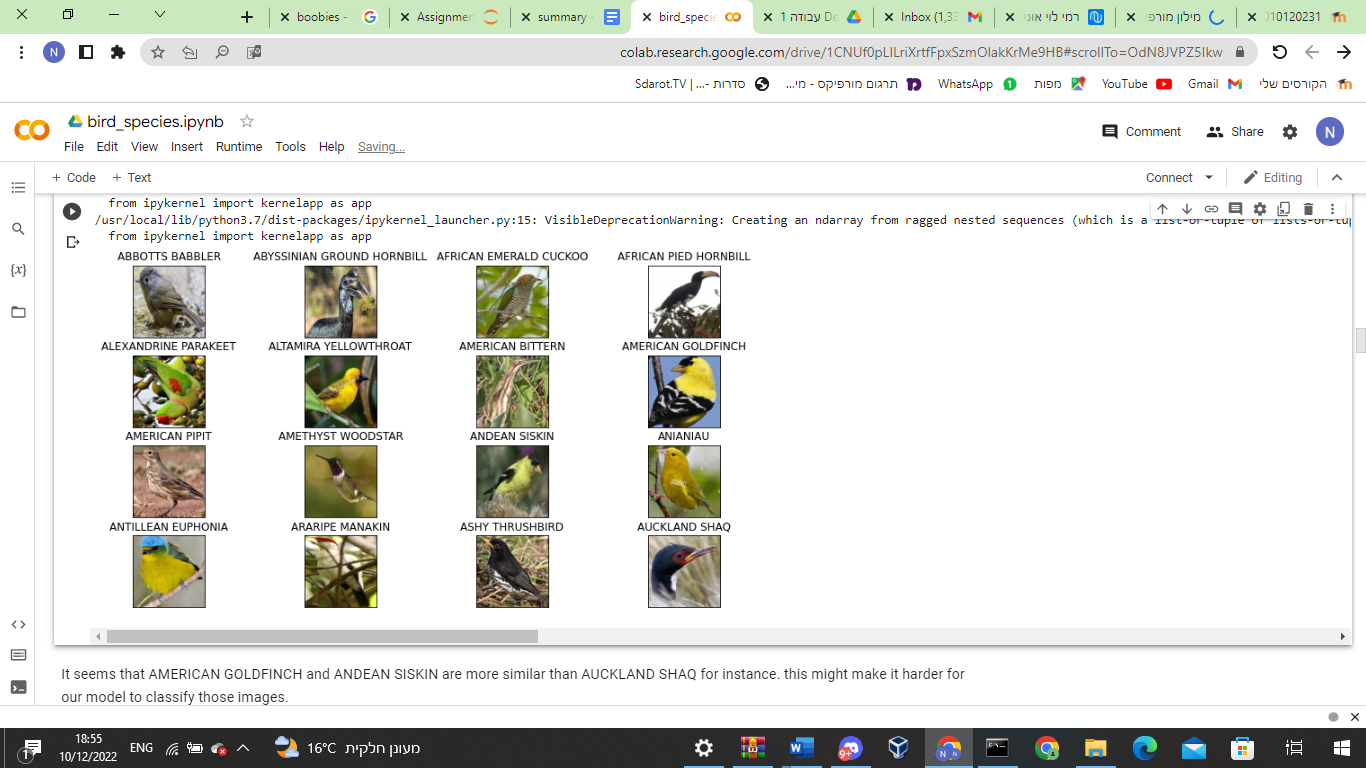
1. After loading the dataset we can see easily that the creator of the dataset made it easier by splitting the dataset into train,test and validation.
2. Each sample is a RGB Image(3 channels) , with the size of 224x224 (written in kaggle, but in any case we will reformat every image to 224x224) and its classification class (number that represents the type of the bird). In addition there are 450 Categories and we created a counter object the maps every class to its # of appearances.



Here is the distribution of our classes in the training set. First we thought about doing data augmentation in order to increase the training set but after consultation and understanding that it won’t benefit us very much, we decided to leave it.

1. As it seems above the data is pretty balanced. there is no need with augmentation for example in order to balance the data, even though the number of instances per class does not show the real distribution.
2. As It seems here <https://www.kaggle.com/datasets/gpiosenka/100-bird-species/code>

There are benchmark results on this dataset with EfficientNetv2(98% accuracy), inception\_v3(Training Data Accuracy: 82.62%, Val Data Accuracy: 88.96%, Test Data Accuracy: 90.50%), And some people made there network from scratch and achieved 71.3% accuracy.



Part 2- Creating our own model:

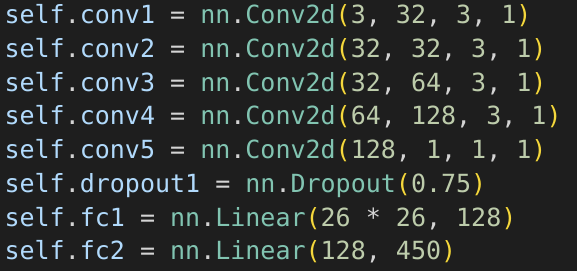
In this section of the project we try to construct our own CNN that will classify an image of a bird to his correct species.

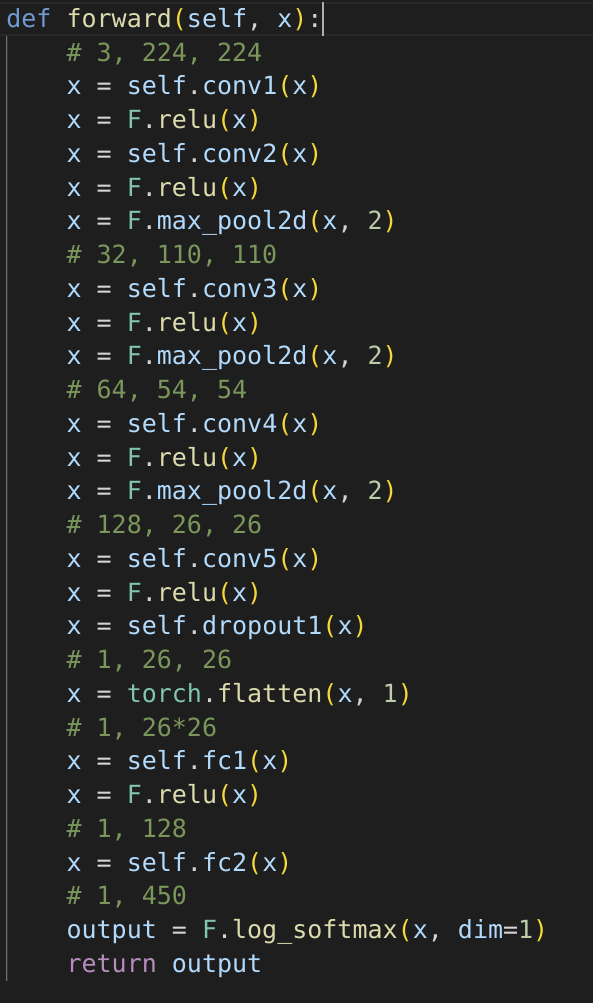
Each model that will represent in the next table was trained with the following parameters:

* Optimizer was adam because in a lot of reviews about why adam is the default optimizer we can see that when we do fewer epochs than 100 adam get better accuracy than sgd optimizer.
* Batch size is 64 because of two things - in lower batch size we can get better accuracy and the ram size of most gpu can’t handle bigger batch size than 512 of our data.
* No augmentation on the whole dataset to avoid leak to the test set.

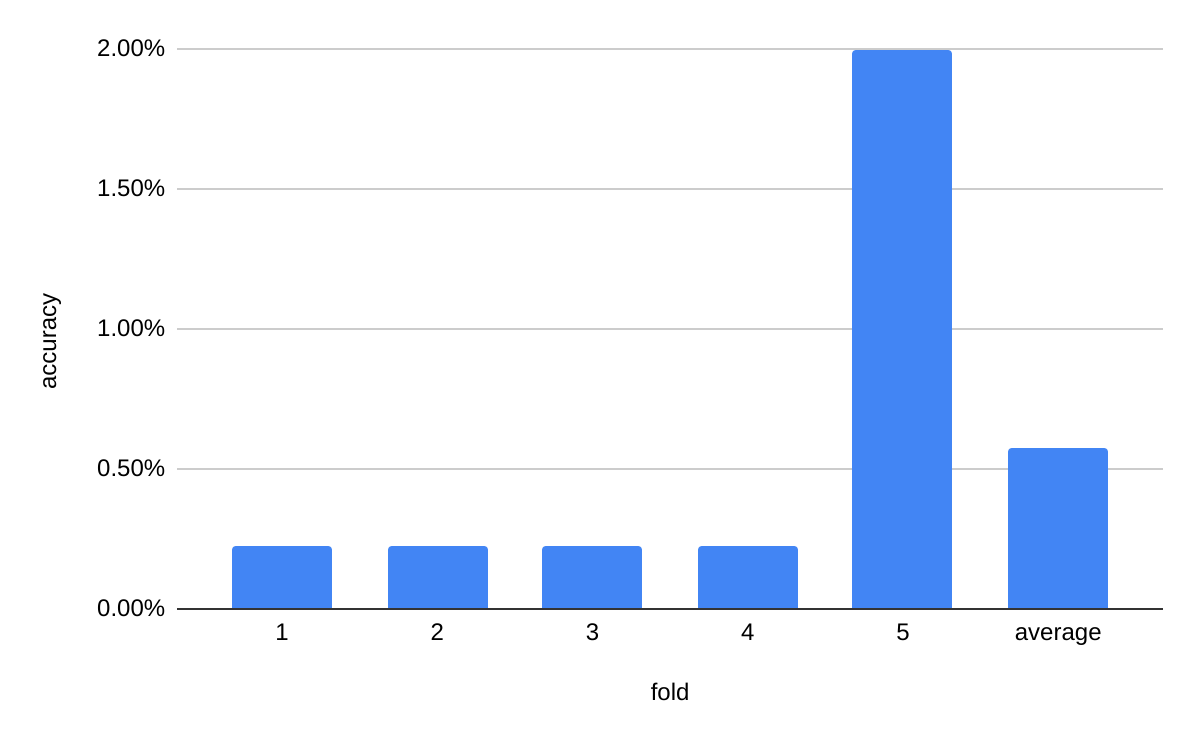
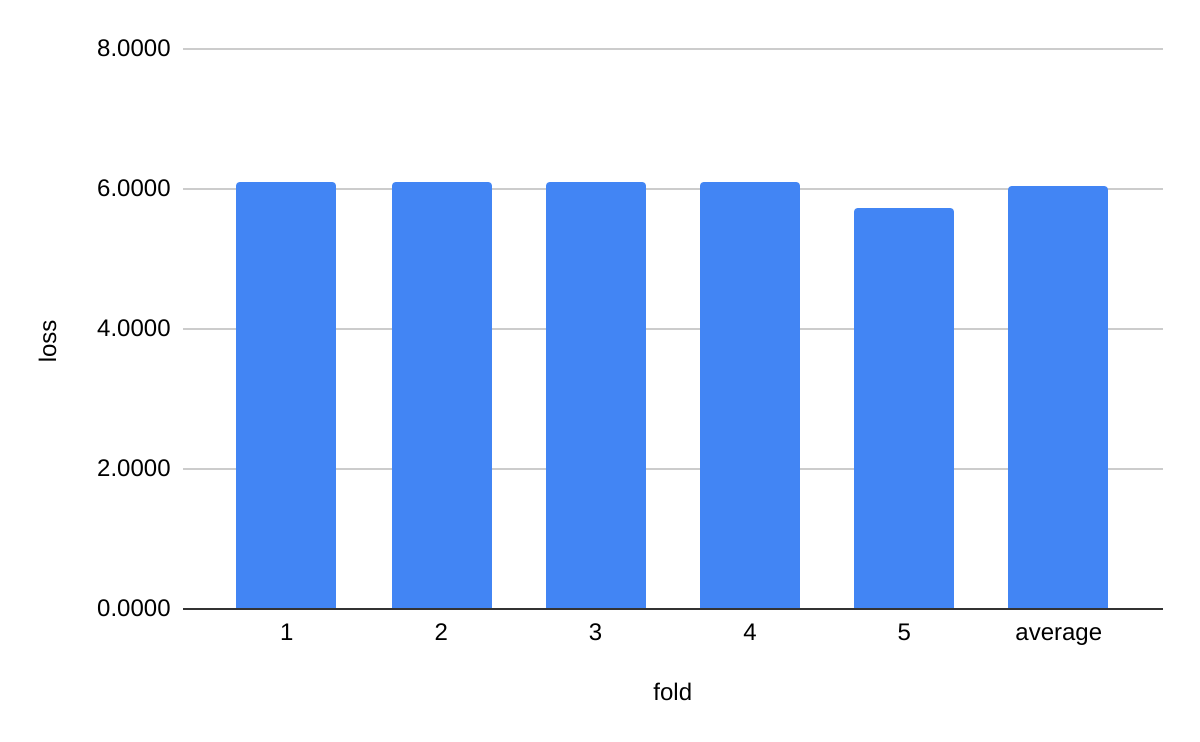
| **data size** | **network** | **learning rate** | **k-fold** | **epochs** | **loss** | **accuracy on validation set** |
| --- | --- | --- | --- | --- | --- | --- |
| 70626 | 1 | 1e-3 | 2 | 10 | 3.0558 | 0.163% |
| 12800 | 1 | 3e-3 | 5 | 20 | 0.5549 | 1.378% |
| 70626 | 1 | 3e-3 | 5 | 10 | 1.2219 | 0.076% |
| 70626 | 1 | 1e-4 | 5 | 50 | 5.764 | 1.250% |

\* Yellow is the net that we will evaluate later  
  
Firstly we used the following net architecture (network 1):



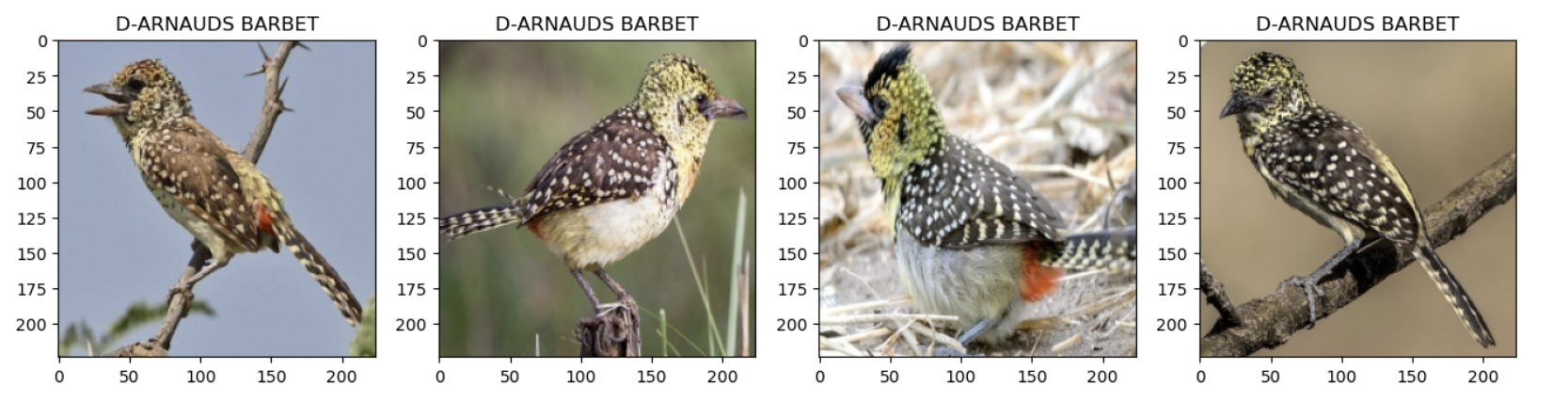
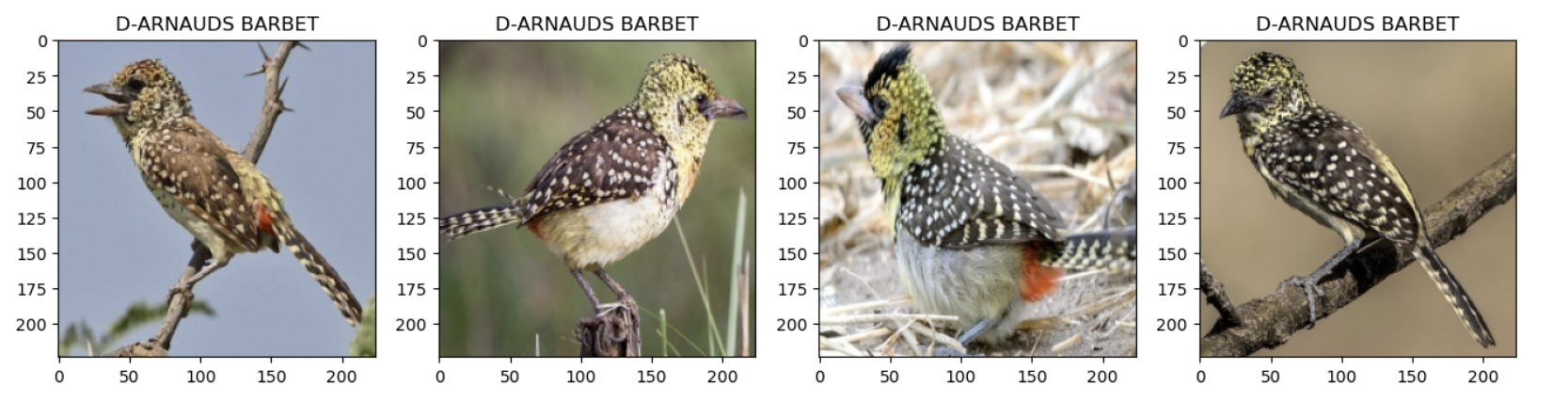


The first thing that we think of was that we got 450 types of birds so we must let the net generalize as it can. We tried to combine two architectures - reduce and embedding. Reduce the problem - pictures that are sized 224x224x3 to smaller problems that we know how to solve ([MNIST](https://en.wikipedia.org/wiki/MNIST_database)) and then try to do something like embedding that let the FCNNs use least information to solve the problem. From comparing our net accuracy to the “regular” net that we saw at class (MNIST that gets bigger size and different output size) we didn’t see any difference so we stuck to our net (this test on smaller size of the dataset was off record so we didn’t save the results).

Net evaluation on testset:

As we can see our model didn’t do very well on the test set. The average accuracy on the test set was worse than on the validation set by almost 50% and in total is still very poor results.   
If we will analysis random algorithm on the validation set we will see that the accuracy would be 0.222% ( 1/450 \* 100). This means that our model is almost 6 times better than random but still is not even close to the 71.3% accuracy that people got on kaggle.

Let's explore one example:



Our model really didn't do well on most of the dataset but this is one of the categories that our model did classify some well, the bird of bread D-ARNAUDS BARBET. He did succeed

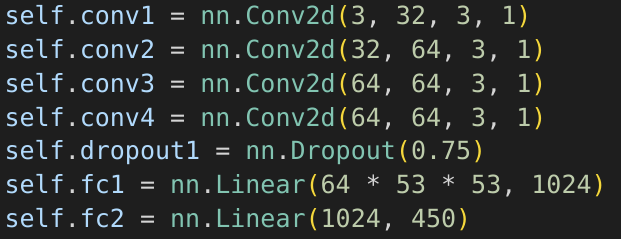
classify the 3 birds on the right but not the one on the left.

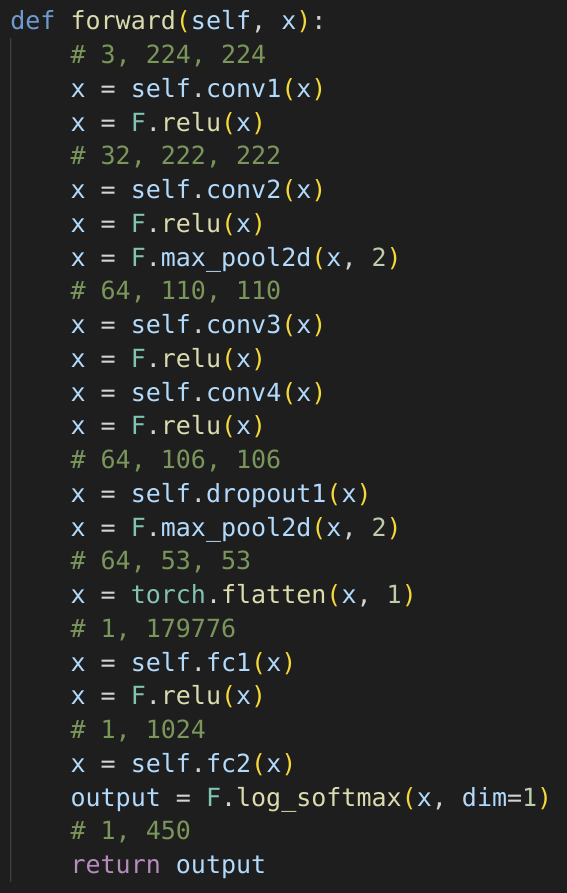
We think the classification is with really low accuracy may result from few things:

* our network architecture is not made well
* our learning rate is getting low faster and the model not learning enough
* we need to train the model for longer

If we look at the result that we got with transfer learning with VGG19 we will see that we got 96% accuracy on the validation set. That means that it can take a long time with our model to get to this result. We decided that we will try new network architecture and decrease the learning rate on the way.

Now we will use the following net architecture (network 2):





| **data size** | **network** | **learning rate** | **k-fold** | **epochs** | **loss** | **accuracy on validation set** | **accuracy on test set** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 70626 | 1 | 1e-4 | 5 | 50 | 5.764 | 0.58% | 1.250% |
| 70626 | 2 | 1e-4 | 5 | 50 | 0.711 | 51.884% | 88.224% |

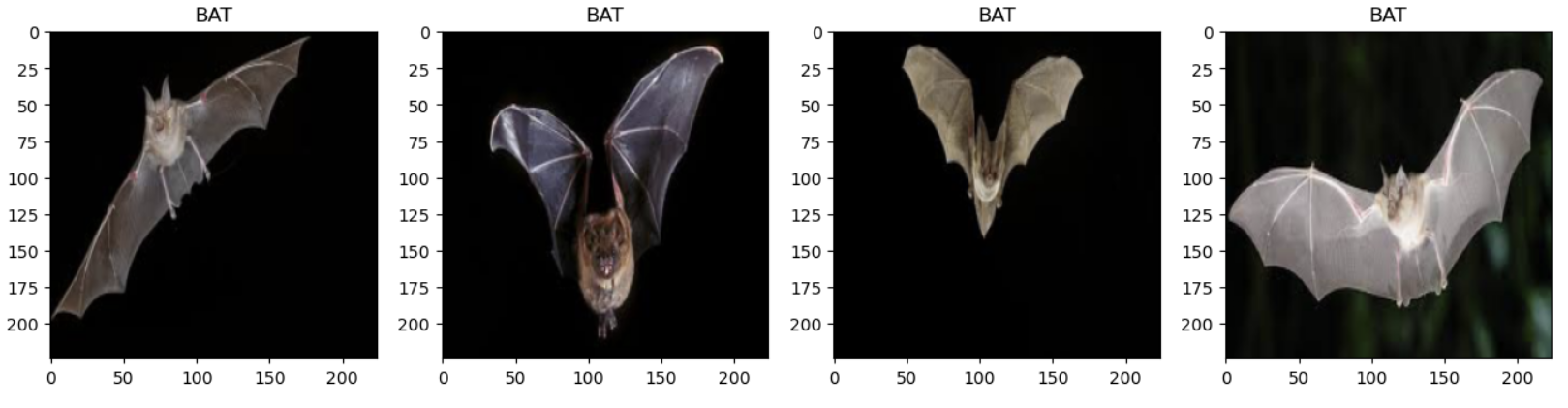
As we can see, the bump of performance is more than x80 in accuracy on test and validation. From that we learned that next time we will come across a new task we first need to try basic network architecture that we can use as a baseline and only then try to do our own tweaks and ideas like we tried first (network 1).

Now we will check our model performance on inference-time-augmentation:

| **network** | **augmentation** | **accuracy on validation set** |
| --- | --- | --- |
| 2 | no | 51.884% |
| 2 | yes | 32.626% |

We can see that our model does not handle augmentation well, to solve that issue we would like next time to add some augmentation in the train set - we think that would help the model to be more accurate too.

Furthermore we trained to train our model on new type of bird - bats:



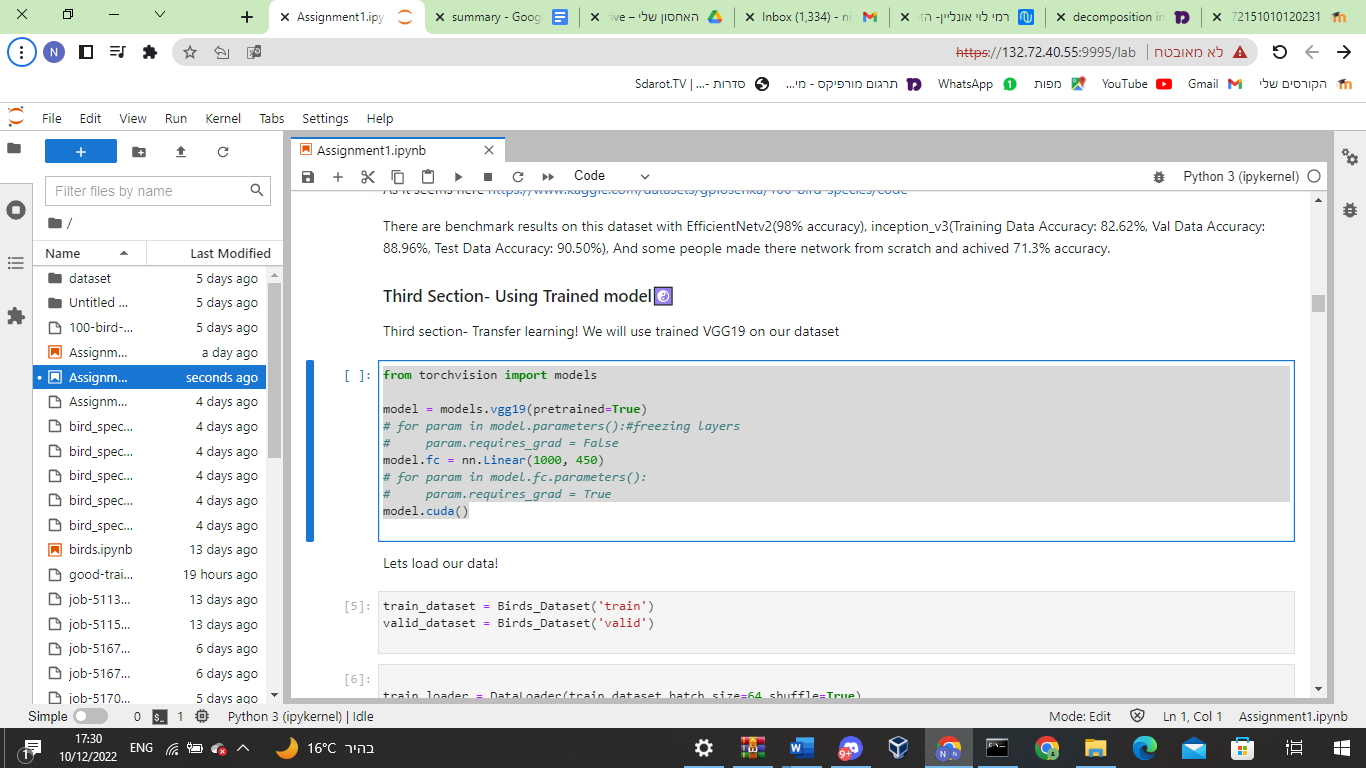
With a success rate of 50% to classify right the label we think that because we needed to add 130 images of bats like in each bird data set and not a few dozen.

Part 3- Transfer learning & Feature extraction for ML model:

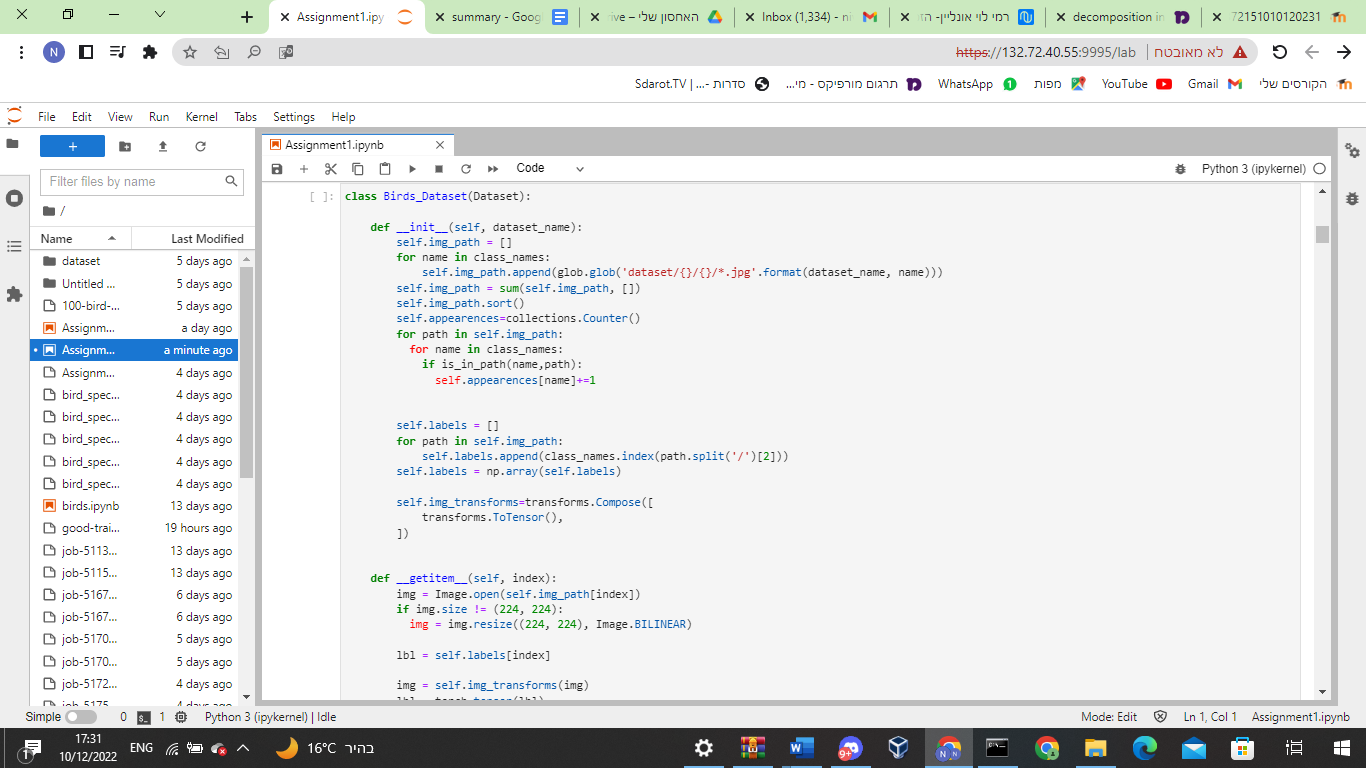
In this part we first of all imported a model from torchvision models. We decided to use VGG19 for our task. Afterwards we added a new layer to the end of the NN so that it will fit our dataset target distribution (450 classes so 450 nodes at the end of the NN).

A.First of all, we imported the relevant VGG19 net, and then added an extra Layer to the fc.

from torchvision import models

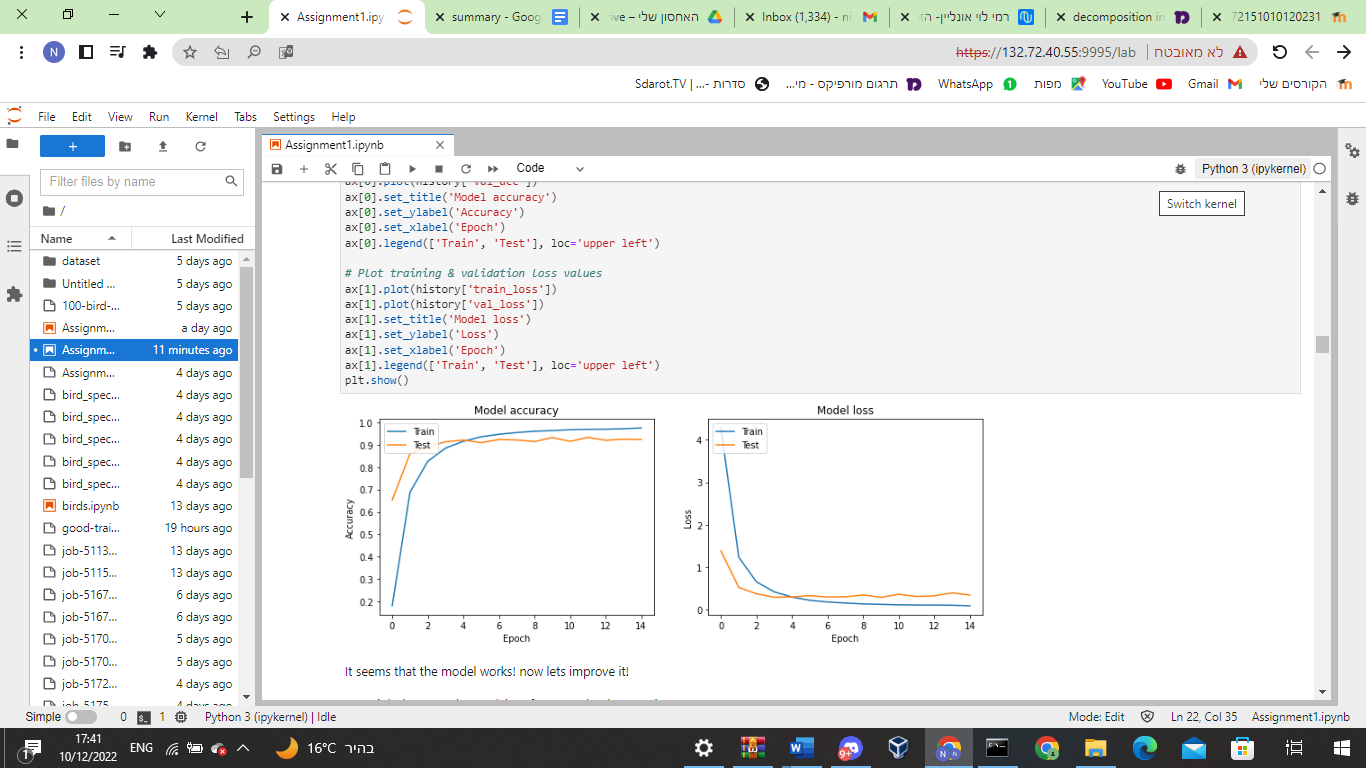


B.VGG19 is already available for pictures that are sized 224x224x3 so we didn't need to change a thing in our dataset class (but even if we should, it would be very easy after creating the dataset class the way we built it).



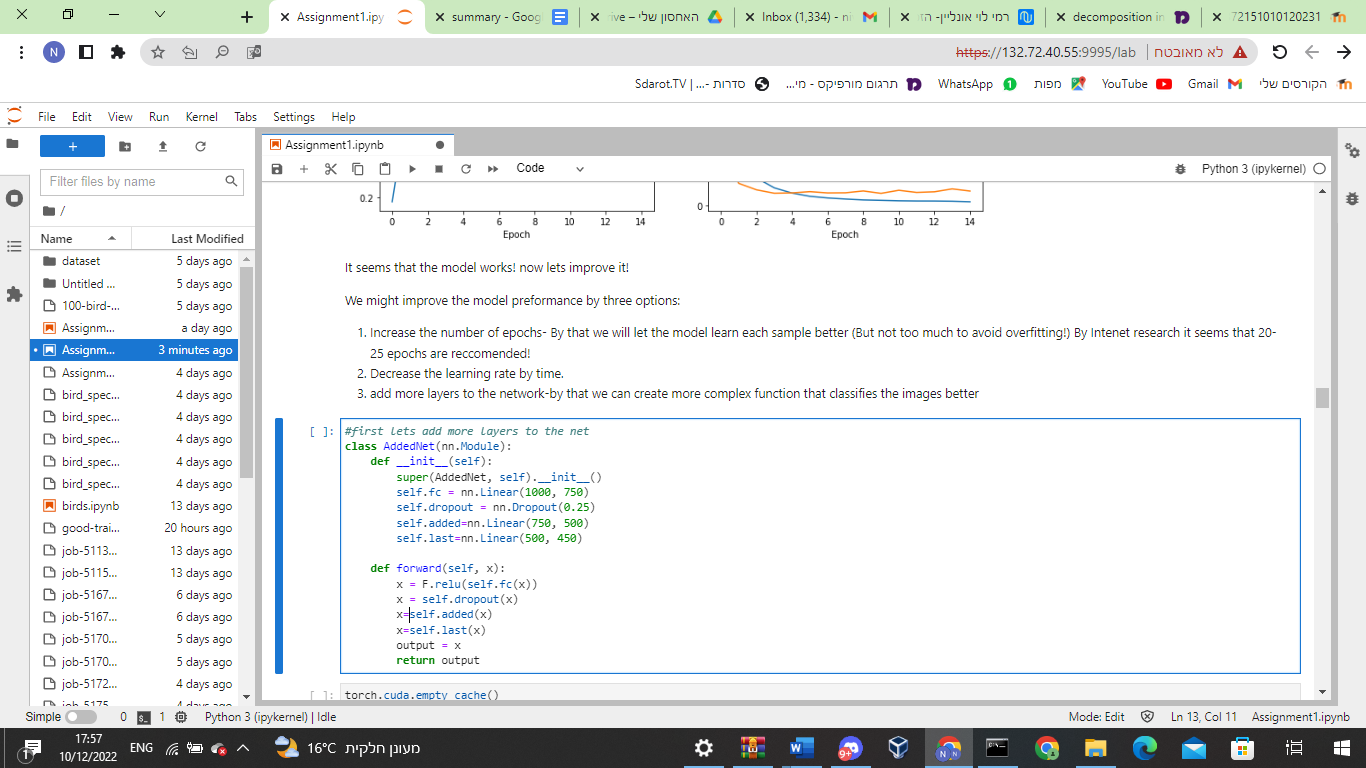
C.In the begging’ the model did not converge no matter how many epochs we did.

Then we changed the learning rate(to 1e-4 from 0.001) and we saw that it really started to converge but we wanted to improve it even more!.

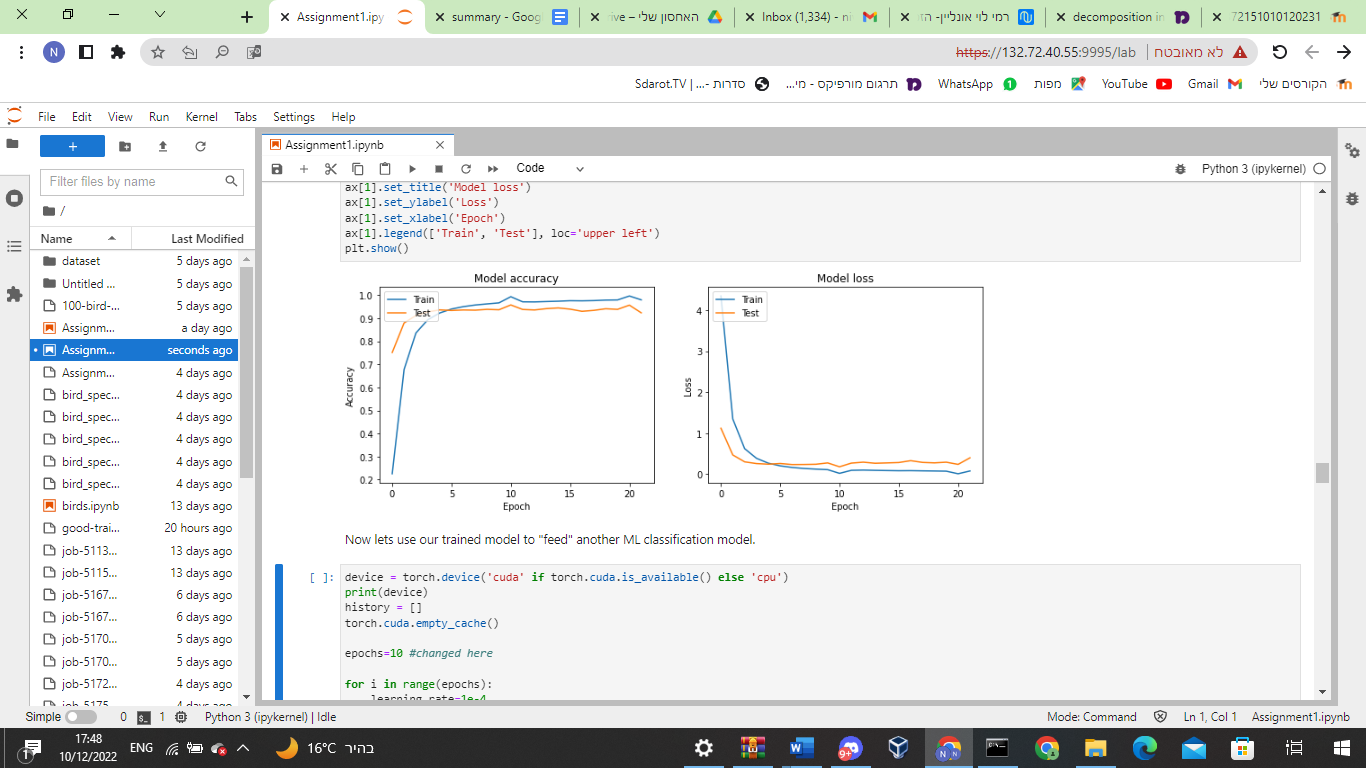


Here we can see that the model was converged(By training loss) and that it knew how to generalize(By validation loss and accuracy).

.Then we tried to use softmax and CrossEntropy function which didn’t converge because it works only on logits not probabilities.Then we decided to add more layers to our network(keep working with the logits, without softmax) train it with decreasing learning rate and more epochs.



We saw that after 10 epochs it crosses the 92% accuracy line (of the validation test) and then we decided that for every 10 epochs we will decrease the learning rate. And it worked! but after 15 epochs we started to see overfitting (the validation loss started to increase while the training loss doesn't really changes)



So we decided that we will try to train in less epochs, and try to reduce the learning rate earlier (After 7 epochs). And it worked!

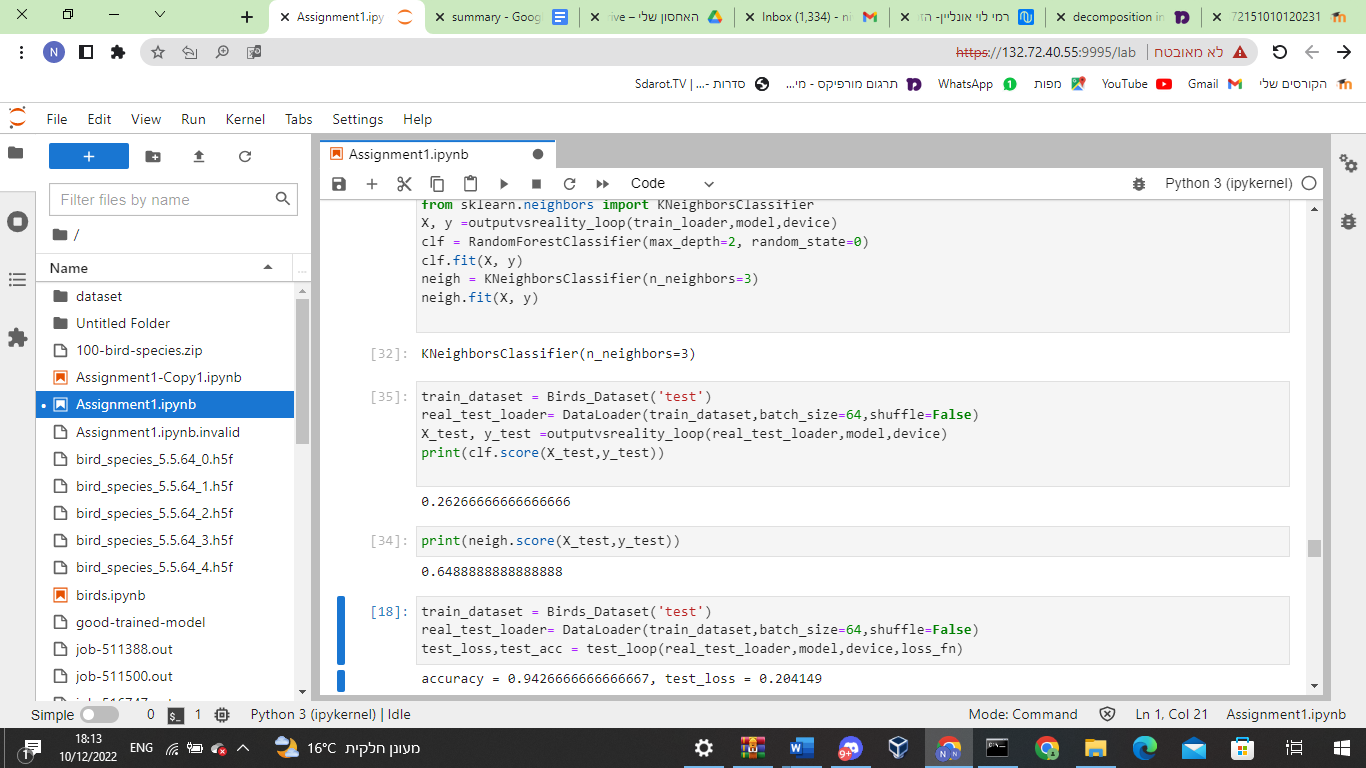


We created a model that has more than 96% accuracy on the validation set and 94.4% accuracy on the test set!.

d. Then we used the model as a feature extractor!, we passed in the model all of the features and saved it in a pandas dataframe. Concurrently, we saved a vector that represent each row classification correspondingly, and then, passed it to 2 ML models:

The first one: Random forest classifier: this model did not show good results after training on the training set (26% on the test set). We understood that because maybe there is a correlation between the features so maybe it is not the best model to use.

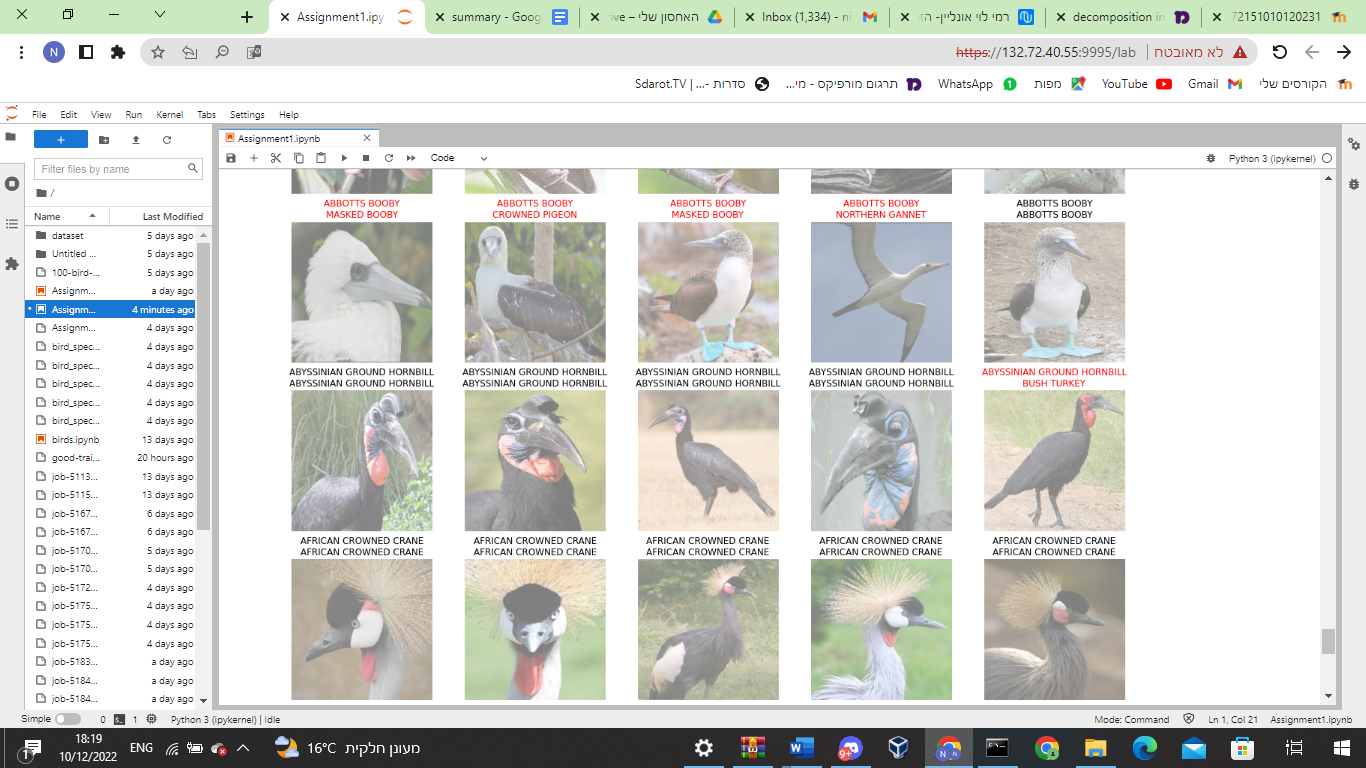
So we tried KNN which gave us much better results (65% accuracy on test set) which makes sense because assuming that the layers will give very similar output to the same bird, a bird with similar output from the test will have a similar representation as a bird from the training set.



Then we did the following experiences:

| data size | network | learning rate | Activation on last layer | epochs | train min loss | loss function | accuracy  (maximum in validation) | reason for failure |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 70626 | VGG+1 layer | 1e-3 | None | 5 | 6.011 | cross Entropy | 0.22% | learning rate too big |
| 70626 | 1VGG+1 layer | 1e-4 | None | 15 | 0.008 | cross Entropy | 92.3% | No failure! but we want better! |
| 70626 | 1VGG+1 layer | 1e-4 | softmax | 5 | 5.998 | cross Entropy | 0.022% | softmax function with cross entropy does not converge |
| 70626 | 1VGG+3 layers+dropout | 1e-4 decreasing | None | 22 | 0.001 | Cross Entropy | 95% | No failure! but might be overfitted by graph |
| 70626 | 1VGG+3 layers+dropout | 1e-4 decreasing | None | 10 | 0.001 | Cross Entropy | 96% | No failure! that is what we wanted! |

When we checked our Improved model on the test we got accuracy of 94.4%.



Here is an example where our model is right for some images but misclassified for others.We can see that there are also mistakes which are very close to the original class (for example it classifies 2 of the boobies as other types of boobies).